DOCUMENT RESUME

ED 207 576 IR 009 692

AUTHOR	Perlman, Radia
TITLE	Using Computer Technology to Provide a Creative Learning Environment for Preschool Children. AI Memo 360.
INSTITUTION	Massachusetts Inst. of Tech., Cambridge. Artificial Intelligence Lab.
SPONS AGENCY	National Science Foundation, Washington, D.C.
REPORT NO	LOGO-24
PUB DATE	May 76
GRANT	NSF-EC40708X
NOTE	32p.
EDRS PRICE	MF01/PC02 Plus Postage.
DESCRIPTORS	*Computer Assisted Instruction; *Computer Programs; *Input Output Devices; *Preschool Children; Problem Solving; *Programing Languages; Research Needs

ABSTRACT

TORTIS is a system of special terminals together with software which is designed to provide programming capability and be accessible for use by very young children. The system is designed to add capabilities in small increments so that the child is never overwhelmed by too much to learn at one time, and maintains a feeling of control over the environment. This system facilitates learning of various concepts such as relative size of numbers, frames of reference, procedures, conditionals, and recursion, but more importantly it teaches good problem solving techniques and a healthy approach to learning. There are improvements which can be made to the TORTIS system; however, certain research must be undertaken before they may be implemented. Twelve references are listed.
(Author/LLS)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U S DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

IR

LOGO MEMO # 24
AIM 360

May 1976

ED207576

USING COMPUTER TECHNOLOGY TO PROVIDE A CREATIVE LEARNING ENVIRONMENT FOR PRESCHOOL CHILDREN

BY

Radia Perlman

Abstract

TORTIS is a system of special terminals together with software which is designed to provide programming capability and be accesible for use by very young children. The system is designed to add capabilities in small increments so that the child is never overwhelmed by too much to learn at one time, and maintains a feeling of control over the environment. This systme facilitates learning of various concepts such as relative size of numbers, frames of reference, procedures, condition als, and recursion, but more importantly it teaches good problem solving techniques and a healthy approach to learning.

IR009692

The work reported in this paper was supported by the National Science Foundation under grant number EC40708X and conducted at the Artificial Intelligence Laboratory Massachusetts Institute of Technology, Cambridge, Massachusetts.

The views and conclusions contained in this paper are those of the author and should not be interpreted as necessarily representing the official policies either expressed or implied of the Naitonal Science Foundation or the United States Government.

Acknowledgments

I wish to thank Dan Hillis, Richard Greenblatt, and Mike Speciner for their hardware help and company during the marathon building and debugging sessions involved in construction of the terminals. Dave Moon and Mike Speciner aided and abetted me in spending an entire week making the characters for the illustrations in this paper. Glenn Wargo painted the original button box, and thought of many of the button designs. He has graciously offered to draw any pictures the computer manages to avoid drawing.

Carl Hewitt's suggestions have been helpful and supportive, and he's been amazingly conscientious about reading and criticizing my papers.

Many of my ideas have been inspired by Seymour Papert's work, and also by Alan Kay's. I enjoyed and learned a great deal from my two summers working with Alan Kay's group at Xerox PARC.

I would really like to thank both Glenn Wargo and Mike Speciner for their friendship over the years. Their support and encouragement have had more to do with any success I have had than they will ever realize.

Table of Contents

1. Introduction 4

2. Why Should Children Communicate With a Computer? 6

3. The Button Box 9

4. The Slot Machine 17

5. What Does the Child Do With the System? 23

6. Ideas for Future Research 26

7. Bibliography 30

Introduction

After watching children learning to communicate with computers, I became convinced of the value of the experience for the child. But I also noticed that for many children, the initial hurdle of typing on a conventional keyboard was very discouraging. I decided to design a new way of interacting with the computer that, although it would not provide the generality of a full computer language, would provide a rich introduction to motivate the children to overcome the typing hurdle, and make many of the advantages provided by the learning of full computer languages accessible to children as young as three or four years.

The TORTIS system consists of two parts. The first part, the Button Box, consists of four button boxes which plug into one another. Each button represents a command. The second part, the Slot Machine, consists of several long rectangular plexiglass rows (each of which represents a procedure) with slots in the top for the user to place cards (which represent commands).

In a system designed for young children it is important that it be simple, since children get discouraged quickly if something seems too confusing or difficult. If it is too simple, however, it will run out of capabilities, bore the child, and have only limited educational value. The TORTIS system deals with this issue by growing with the child. The system is designed modularly, so that the child starts out learning a simple subset of the commands and more can be added without disturbing the part of the environment that the child feels comfortable with. More commands are added by plugging a new button box with a few buttons into the set of button boxes the child is already familiar with (if the child is playing with the button box part of the system) or by giving the child more cards (if he is playing with the Slot Machine). This way the child really has a feeling of control over the

environment, since if he gets overwhelmed by the new part of the system, he can unplug the offending box or remove the new cards, physically remove them from sight, and concentrate for a while on the part of the system he knows. Later, when he is feeling more adventurous, he can retrieve the newest part of the system.

While upward compatibility is an important part of the system, (because the child feels comfortable if he knows that the part of the system he understands still works the same way he is used to), there is a trade-off between upward compatibility and ideal behavior of the system at different stages. The TORTIS system is, for this reason, not completely upward compatible. The most obvious transition point is from the Button Box to the Slot Machine, where everything changes, but there is a small incompatibility between how the Button Box system behaves with the most advanced add-on box, and without it. I felt this was necessary and not overly traumatic for the child, especially because the first time the environment is disrupted by changing its behavior, the child is already adept at and comfortable with the system.

Another design issue is how easy the system should make it for the child to achieve interesting effects. If it is too hard, the child will become bored with doing rather mundane projects and will become discouraged if he tries anything harder. If it is too easy to quickly achieve spectacular results, the system will become "magic", where the child follows a rote procedure he does not understand, he will not really feel in control of the environment, and he will not learn as much since he will not have to think. With the Button Box system the children quickly learn the function of each button, but it is a little too hard for them to do anything complicated. This is not a serious problem since they will graduate to the Slot Machine before they become bored with simple projects. With the Slot Machine it is, if anything, too easy for the child to achieve spectacular results.

Why Should Children Communicate With a Computer?

To begin with, it is fun. A computer is a toy that never runs out of capabilities. The child's first experience with the system involves talking to the "turtle", a small (12 inch diameter) circular computer-controlled robot equipped with a light, horn, and pen, and capable of moving in a straight line in the direction the light is pointing, or rotating about its center (where the pen is). The pen can either be up (off the paper), in which case the turtle does not leave a trace when it moves, or down (touching the paper) in which case the turtle draws a line as it moves forward or backward. By expressing himself clearly in the TORTIS language the child can drive the turtle around the room, drive the residents of nearby offices out of their rooms (by tooting the horn over and over), draw pictures on the paper or the floor, knock over piles of blocks scattered around the room, play music, play tag with another child and turtle, teach the turtle how to negotiate a maze, or concentrate on the indirect consequences of commands such as the pictures of the commands that get displayed on a screen when buttons are pushed or the lights that flash when the Slot Machine is operating.

Since learning TORTIS is fun, easy, and not the kind of activity at which he can be judged as failing, (there are no error messages because every command is understood by the system and the child can proceed at his own pace) the child should develop a healthy attitude towards learning.

More specifically, there are many new concepts the child can learn with this system. Since the turtle only understands the commands "forward" and "turn" and not commands of the form "go that way", the child has to learn about frames of reference and how to translate from one frame to another. Wanting to tell the turtle which way to turn will show

the child the importance of learning the concepts "right" and "left". When the child starts using numbers, he discovers that there are small and large numbers. Many children know how to count and know how to say the alphabet, but don't realize that it is more significant that "2" comes before "7" than that "B" comes before "G".

In order to solve the simple problem of making the turtle go in a direction other than the one it is pointing, the child has to break the problem down into smaller, solvable steps, a valuable problem-solving technique. This technique is even more important in larger-sized problems, like the problem of drawing a house or one's own name.

In order to draw pictures the child will learn geometry, since he will have to discover what angles the turtle must turn in order to produce the desired effect. He may discover, especially with help, such facts as that the sum of amounts the turtle turns in drawing a polygon-type picture is constant (a valuable fact to know if one wants to calculate how far to turn in drawing a regular polygon with a given number of sides).

The concept that one can name a set of commands and use that list of commands over and over is valuable in learning and in expressing oneself. To figure out how to draw a flower, for instance, the child will usually discover that it is easier to teach the turtle to draw a petal, name the set of commands that cause the turtle to draw the petal, and use the new command in the process of drawing a flower.

The concept of conditionals is easily acquired by "teaching the turtle" (writing a procedure) to walk forward until it hits the wall, in which case it should stop. (The turtle is equipped with a touch sensor and it can tell the computer whether it is touching something).

Another advanced concept that may be accessible to young children is the concept of variables, quantities that can change in value. If one uses a variable as the distance the

turtle should travel forward, one can change the size of the picture the turtle draws by varying the value of the variable.

It requires a deeper understanding of a problem to explain how to do it to a computer than to do it oneself. If the child teaches the computer how to display a clock showing a given time, or better yet, display the current time (it is possible to find out from the system what time it is) the child will have a more thorough understanding of how to tell time than he could probably acquire any other way.

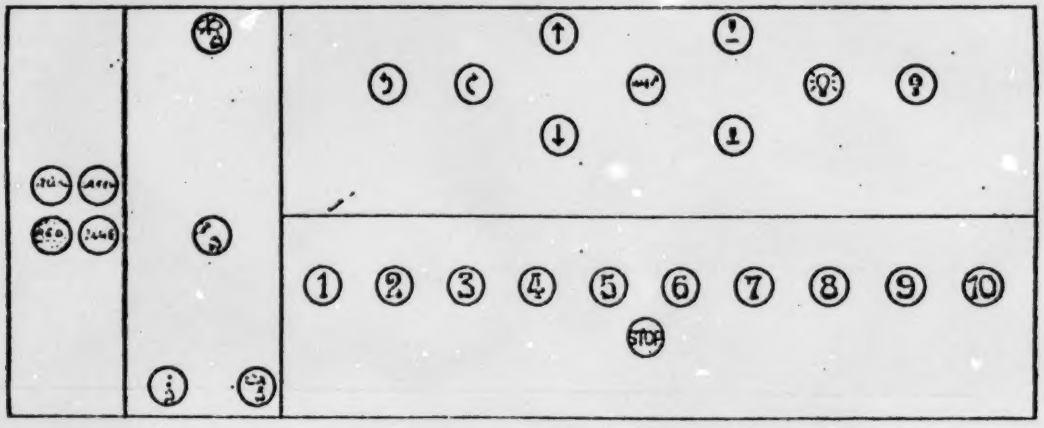
Drill and practice programs are occasionally useful for learning something like arithmetic, however it is more fun to use such a program if one writes it oneself. Drill and practice programs are easy to write, and the child will certainly play with any program he writes. In ordinary computer aided instruction the child has little control over the environment, and any of the worthwhile aspects of the CAI environment can be duplicated by the child in an environment the child controls.

The most important thing the child learns in an imaginative computer environment is a fearless, joyful attitude towards learning. Very few programs run correctly the first time. One of the most enjoyable activities one engages in when communicating with a computer is debugging, finding out why the computer misunderstands what you want it to do. An answer on a multiple choice test is either right or wrong, but the word "wrong" is meaningless when applied to a computer program. The word "right" is almost meaningless too, since most programs can be improved or made fancier. The concept of debugging can be applied to any skill the child wishes to learn. Eventually the concept of "failing" at an activity will be meaningless to the child. If he is not doing the activity as he wants, he will try to discover where the bugs are and correct them, instead of ever giving up.

The Button Box

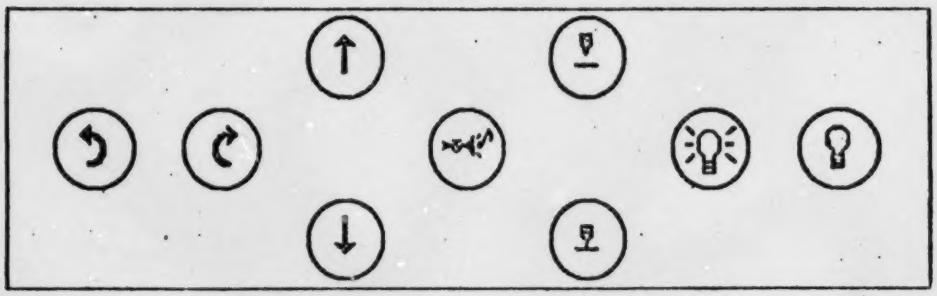
The Button Box is the first part of the TORTIS environment to which a child is introduced. It is actually several different boxes of buttons which plug into each other. The design philosophy is that the child is presented with only a few new buttons at a time, and (except at the beginning of course) there is always a set of buttons which the child really understands. If he ever feels overwhelmed by the new buttons, he can remove them from sight (which the children often do) by unplugging the latest box and hiding it, and play with the part of the environment he feels he has complete control over. When he wants to learn something new he can get the next boxful of buttons, with the comforting thought that if he gets confused he can restore the state of the world.

the complete button box:



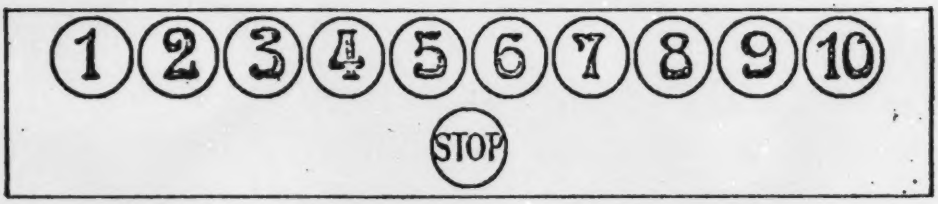
The first box of buttons is the "action box". Every button corresponds to a possible action the turtle can perform and when a button is pushed the turtle immediately responds. The buttons are "forward", "backward", "rotate clockwise (right)", "rotate counterclockwise (left)", "toot your horn", "put the pen down onto the paper", "lift the pen off the paper", "turn your light on", and "turn your light off". The number of degrees the turtle turns each time "right" or "left" is pushed can be set by the teacher when the system is started. I will assume in this paper that each turn is 5 degrees.

the action box:



The next box is the "number box" and adds buttons for the numbers from 1 to 10, as well as a "stop" button which interrupts the action of the turtle. Pushing a number before an action causes the action to get performed that many times. If a number is not pushed the turtle performs as before, doing the action one time.

the number box:



The third box is the "memory box". It has four buttons, "start remembering", "stop remembering", "do it", and "forget it". The memory box allows a set of commands to be stored and re-executed. It is used with a display, so that the child can watch as each command gets stored, and can refer to the list as it gets executed.

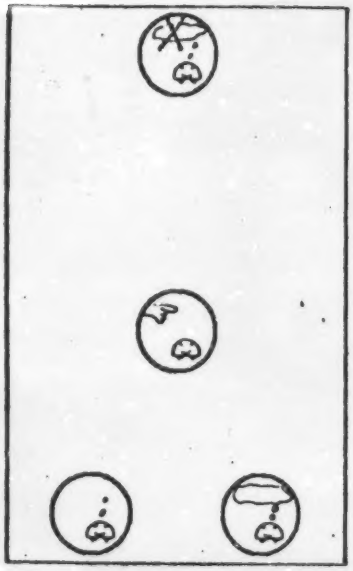
When "start remembering" is pushed, the display screen lights up in anticipation of direct commands. When the child pushes a button, the picture on the button is listed on the screen, and the command is executed.

Pushing "stop remembering" causes a line to be drawn below all the listed commands, indicating nothing further will be written there (though more can be added by pushing "start remembering" again).

Pushing "do it" causes the entire list of commands on the screen to be executed in order. A box is drawn around the command currently being executed so the child can follow along. "Do it" can be used with a number so that "3 do it" would cause the screen commands to be executed 3 times.

Pushing "forget it" clears the screen.

the memory box:



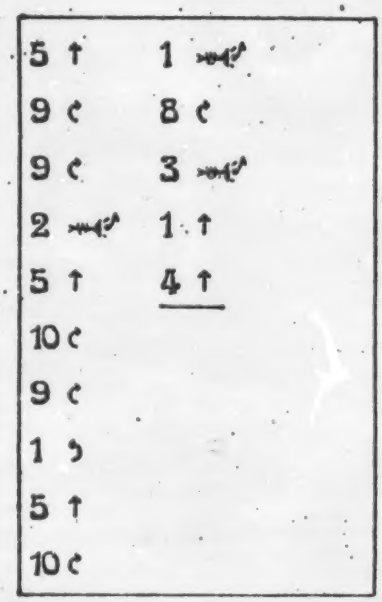
the four procedure box:

The fourth and last of the currently implemented button boxes is the "four procedure box" which adds four colored buttons, each to a quadrant of a screen. Along the boundary of the screen each quadrant is colored like the corresponding button. Now "start remembering", "do it", and "forget it" require a color button afterwards, indicating which quadrant of the screen is to written on, executed, or erased. Since "do red" is a valid command, it can be used as a command in any of the four procedures, including itself, so this box allows subprocedures.

Example: What the screen will display when programmed with and executing a procedure to draw a square, with the turtle stopping to toot occasionally.

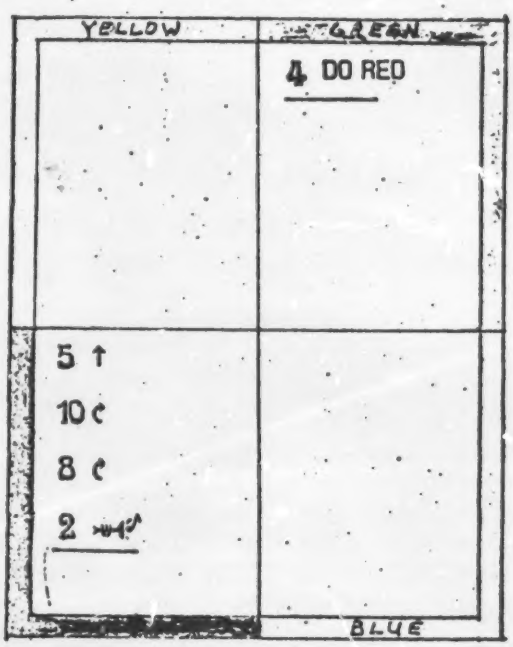
assuming single procedure

box used



assuming four procedure

box used



I used the button box system with around 25 children aged 3 to 5. I learned several interesting things.

1) Once a child is confused by something he gets very discouraged and says, "that is too hard for me" or "I'm not smart enough for that", both of which are certainly attitudes we do not want to cause, so it is important not to give the child new buttons before he is ready.

2) Some children require constant interaction and suggestions about further things to try or else they start doing one thing over and over and get bored.

3) The "toot" button is their favorite button.

4) Many children do not look at the pictures on the buttons and discover to their amazement after using the button box quite adeptly for several sessions that the pictures on the buttons have something to do with what action the button causes.

5) Most children have trouble with "right" and "left". Most of the time the children push one of the two at random and if it is wrong then push the other one. Seymour Papert discovered a trick the children can use to get the correct command. The child looks at the turtle, decides how he wants it to turn and moves his finger in a circle in the desired direction, keeps his finger moving wh'' moving it over to the button, and matches it with the proper curved arrow. There is more motivation to get "forward" and "backward" correct the first time because they draw lines and can mess up a picture. Most children eventually adapt to the turtle's frame of reference for "forward" and "backward" though almost all are surprised the first time they push "forward" with the turtle facing them.

The most important observation was that for four-year-olds the conceptual leap to the memory box is too large. Several four year old children did manage to use the memory box but they never seemed to completely understand it. There were three possible ways

that they used it.

1) They understood that pushing "start remembering" caused pictures to get displayed on the screen, much to their delight. They would push "start remembering", fill the screen with commands, and then push "forget it" to clear the screen. While playing with the display they ignored the turtle completely, as though the two things were completely unrelated.

2) They pushed "start remembering", and then, ignoring that anything different was happening, drew a picture. Then they hit "do it" to get the same picture again. If when drawing the picture they had, for instance, made the turtle turn the wrong way at first and then corrected, or stopped to make the turtle toot several times, they were surprised that the turtle did the same thing when drawing the picture the second time. In other words they thought the picture was the procedure as opposed to the set of commands the turtle executed while remembering. While the children were delighted by all the stuff that got displayed on the screen they did not realize all those pictures were commands and steps in the procedure. Even executing the procedure in "slow mode", where the turtle would wait a certain amount of time between steps so that the child could more readily follow the action, did not cause the child to understand. Probably the turtle, the display, and the button box are too many things to keep track of at once.

3) They pushed some moving-buttons randomly and then pushed "10 do it" many times, realizing that that caused pretty patterns. While this was a lot of fun, it was more a magic mechanical process that somehow created nice results as opposed to being something the child knew he was really controlling.

In addition to the problem that the jump to the memory box is too large a conceptual leap, the button box has the additional disadvantage that there is no way to edit

procedures. Without the ability of easy editing the child does not learn the healthy concept that there is no "right" or "wrong", and that any procedure can be fixed to do what they want, and even a working procedure can always be improved. But adding editing commands before the child has mastered the four memory buttons would overwhelm the child with too many new concepts. To fix these problems I devised the Slot Machine.

The Slot Machine

The Slot Machine consists of several long rectangular boxes, called rows. Each row represents a procedure. On top of the row are slots into which plastic cards the size of credit cards can be placed. Each card has a picture on it, representing a command. At the left end of each row is a button. Pushing that button causes the commands represented by the cards to get executed in order. A light in front of each slot lights up when the card in that slot is being executed. Each row is colored a different color. A solid-colored card represents a subroutine call to the row of that color. For instance, a red card means, "do everything in the red row".

The advantage of the Slot Machine over the memory box of the Button Box system is that commands are stored by the child, not by some magic that occurs when it somehow enters a different mode. And the problem of editing goes away completely. If the child wishes to move a command, or change it, he simply moves or removes the corresponding card.

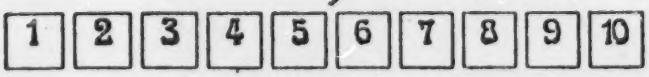
The Slot Machine is a very general purpose device. The language implemented on it is easily changed around and experimented with by using a different set of cards (cards are cheap and easy to make) and modifying the program that runs it. For instance, Jean Bamberger's music system is easy to implement. Each card can represent a tune block. Rows can even be run simultaneously so that two tunes can be played in harmony.

The Slot Machine has the same advantage that the button box system does, in terms of adding to the language in reasonable increments. The child can start out with a small set of cards, and be given more cards when he is ready. He doesn't have to use the new cards if he starts feeling insecure about not understanding the new part of the system.

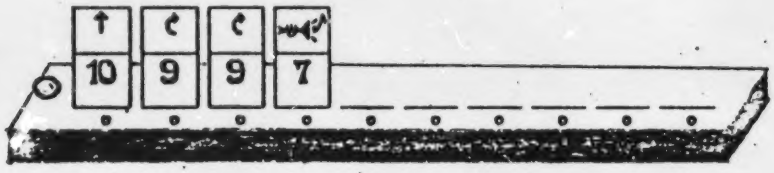
I think it is important that each slot contain room for an entire command, i.e. I would not be happy with the child having to place part of the command, say a conditional, in one slot, and the rest of the command in the next slot. In the language I envision for the Slot Machine (only part of it is implemented; the rest will be implemented after more is observed about how children react to what is there), there will be three different kinds of cards--number cards, action cards, and conditional cards. The three kinds of cards will have different heights so that all three pictures can be read when all three cards are present in a slot. In the basic language (what is currently implemented) there are action cards (picture cards for the turtle functions, solid colored cards for subroutine calls, and end of procedure cards) and number cards.

action cards:

number cards:



Example: A Slot Machine program to draw a square and toot.

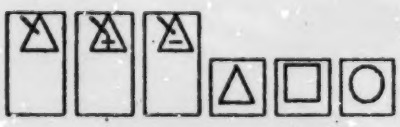


Since a display is not needed to display a listing of programs the child has made (as is done with the button box), the display can be used to draw pictures instead of having the turtle draw them on the floor. The "display turtle" is a triangle displayed on a screen which obeys the same commands as the floor turtle (except it doesn't have a light or horn and it understands CLEARSCREEN). The display turtle is faster and more accurate, and if the child wants to concentrate on drawing pictures (as opposed to knocking over piles of blocks or driving the turtle through a maze) the display turtle is much more reasonable to work with.

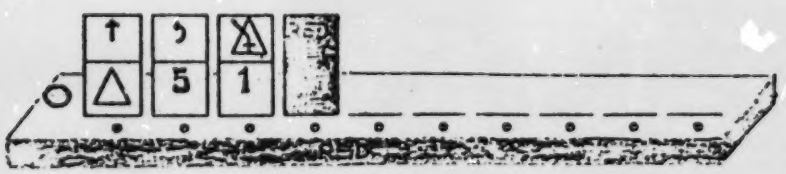
If the children have no problems with learning the basic language, variables will be added. When the children use variables, an extra display may be used for the variables, or a portion of the turtle display can be reserved for them. Or there can be a special purpose box with digital readout for each variable.

The way variables will be implemented is there will be several (3-5) different shaped polygons displayed with numbers inside. There will be cards the size of number cards with each polygon drawn on it. Thus the command "triangle toot" would make the turtle toot its horn the number of times indicated inside the triangle on the screen. There will be action cards with a picture of a pencil writing into one of the polygons. Thus the command "write into triangle" writes the number on the number card, or "1" if no number card is used, into the triangle variable. In addition, an action card with a picture of a triangle with a "plus" inside means "add the number to the value inside triangle". With variables spirals are easy to program, numbers bigger than "10" are possible, and arguments can be passed by setting the value of a variable before calling a procedure that uses it.

examples of variable cards, action and number cards:

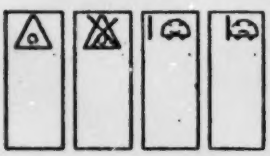


Example: A Slot Machine program which draws a spiral.

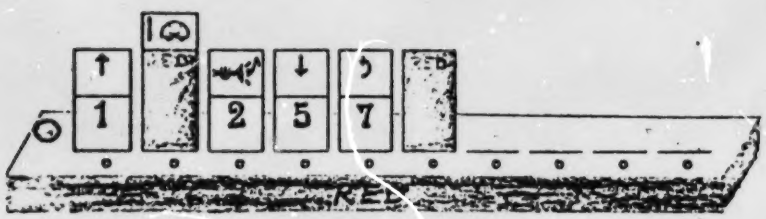


If the children still have no problems I will try adding conditionals. There will be conditional cards for "if the turtle is hitting something", "if the turtle is not hitting anything", "if triangle is 0", and "if triangle is not 0" (and similar ones for the other variables). Adding a conditional card to the command in a slot means that that command should be executed if the conditional is true.

examples of conditional cards:



Example: A Slot Machine program which has the turtle toot and walk in a different direction when it hits something.



In working with the Slot Machine with children aged 4 and up, I have discovered that it is so easy for them to make pretty pictures by placing a random collection of cards into the rows that they seldom plan a procedure. The children understand what the system is doing, because they can explain what is happening, and they can "write" a procedure that is specified by someone else, but they will not acquire concepts like subprocedures unless they discover the usefulness of the concept in projects they are doing for themselves. The ease of writing and modifying programs with the Slot Machine is therefore almost a drawback. Also, the size limitation of the Slot Machine limits the kinds of projects the child can undertake (for instance it is probably too small to allow a child to write a proceduro to draw the child's name on the screen). For this and other reasons it is important to design a full programming language together with a highly interactive environment that would be accessible to young children to use after the Slot Machine.

What Does the Child Do With the System?

Now I will relate a typical scenario of a child's interaction with the system over several sessions. Each session is about an hour long, and the child usually comes once a week for about ten weeks. This isn't a diary of a specific child, but instead is a description of what seems to be typical, based on experiences with many children.

The first thing the child does is inspect the turtle. Then the teacher suggests that the child push some buttons and see what they do. The child does this for a while, and the teacher asks the child to verbalize what each button does. It is typical for the child to say that all four buttons FORWARD, BACKWARDS, RIGHT, and LEFT, "make the turtle move". The teacher asks about how each of those buttons differ.

When the child is reasonably comfortable with the box and knows the function of each of the buttons, the teacher starts to suggest projects, like moving the turtle to different points around the room, knocking over a pile of blocks placed somewhere, or parking the turtle under a chair. When the child seems able to control the turtle, the teacher can suggest the child try to draw pictures with the turtle, like a square or a letter. Sometimes the teacher prepares a "connect the dots" picture for the child to play with, so that the child can have a surprise picture when he is done.

When the child is comfortable with the beginning button box, the number buttons can be added. Usually they are introduced with the "toot" key, and the child spends a long time pushing various numbers and counting along to keep the turtle honest. Usually in using numbers with the motion keys, at first the child either hits "1" (if he wants a small number) or "10" (if he wants a large number). When the child starts drawing various projects he usually starts using the other numbers too.

At this point the child is ready for the Slot Machine. First the teacher demonstrates placing a single card in the first slot position of a row (and leaves the other rows hidden) and pushing the button to start execution of the row. The child replaces the card with other command cards and pushes the button. Then the teacher explains about the light in front of each slot position, and by using a toot card, and placing it in different locations in the row, shows how the turtle does the command when the light gets up to the position where the card is. The child plays with it for awhile, predicting when the turtle will toot. Then the teacher suggests the child try putting two toot cards somewhere in the row, and the child notices how both of them get executed. Then the teacher suggests the child can put different cards in the row and they will all get executed. (Most children do not need this much prompting. It is best if the child works completely independently, but the teacher is there to explain things if they get confusing, or suggest projects if the child cannot think of something to do.)

At some point the child will switch over to using the display turtle, since it is too difficult and unrewarding to make pictures with the real turtle, which is slow and inaccurate, and which obscures the picture it is drawing. The display turtle is hard to understand if presented at the beginning, but once the child understands the other turtle, especially PENUP and PENDOWN, the switch to a display turtle is not traumatic.

Next the child is given number cards, and if he already understands numbers on the button box, he will use them easily. The procedure cards are a little more confusing and have to be explained. Various games can be played with procedures, like having the green procedure consist of 4 TOOT, and having the red procedure consist of 2 GREEN, and asking the child to predict how many times the turtle will toot. In this way multiplication is a useful and understandable concept. Recursion seems to be a confusing concept. When the

children program the green procedure to include a GREEN card, for instance, and after GREEN has been executed inside of the green procedure a few times, if the child removes the GREEN card the Slot Machine will "pop its stack", executing what is left of the green procedure for each time GREEN was called. The children do not understand this, and usually just ignore it.

After a child is this far advanced it is not unreasonable to start on a more complete language like Logo, especially if the child knows the Slot Machine and Button Box are available to be played with (in case he gets overwhelmed for a while). And especially if the traditional programming environment is improved with display editors, display debuggers, and pointing devices, the transition to it from the Slot Machine can be comfortable.

Ideas for Future Research

There are certainly improvements that can be made to both the Button Box and the Slot Machine. Children as young as the ones that use this system are not particularly interested in drawing pictures. Music seems like it might be a good activity for children that age. There are different ways of implementing music for very young children. One obviously bad way is to force them to type in individual notes in something close to musical notation. A better way is to use the Slot Machine, and have command cards for each note, with the number argument used as a duration, or to have each card be a "tune block" as used by Jeanne Bamberger. Another possibility is to have the child "draw" the tune on a display using a pointing device, where the vertical position of a line indicates which pitch is to be played. When the tune is to be played, the computer will scan across the screen at an even rate of speed, playing whatever pitches have points at the horizontal position of the scan. Pitches can be quantized, so that any point within a certain interval corresponds to a certain note, or frequency can vary continuously with vertical position, which would make it harder to write specific tunes but would produce interesting effects.

The Slot Machine has the problem that initially there is so much that the child has to do to get the turtle to execute a command (find a card, put it in a position, and push a button) that it is sometimes difficult for them to understand that each card stands for some specific command. An improvement to the Slot Machine, which might even make the Button Box completely unnecessary, would be to have the light in front of each slot position be a lightable pushbutton instead, such that pushing one of the slot pushbuttons would cause only the command in that slot position to get executed, but pushing the large button on the left of a row would still cause the entire row to get executed. This would also enable the

child to debug a program, since the child can cause it to get executed arbitrarily slowly by pushing each position in turn.

The Slot Machine is also physically too large, and does not have enough information in each position to really allow numbers, commands, and conditionals. The size was determined by the hardware design, especially the type of sensors used to read the card. A relatively simple hardware design change would allow it to be smaller and contain more bits of information in each position. Also, if the Slot Machine could be designed to be reasonably inexpensive, many places would like to use it.

TORTIS is not a complete programming language. The children will soon get limited by the finite number of procedures they can write, the finite length of the procedures, and the small size of numbers. Once they have mastered all the concepts in this system they are sophisticated enough to use a more complete language, even if they are still physically or psychologically incapable of coping with the standard keyboard-oriented languages. The standard programming environment should be modified to accommodate them, and some of the ideas will be useful for all users of the environment.

A simple improvement to the standard programming environment, which cuts down on the amount of typing necessary, is to use a "recognize" key, as implemented in the timesharing system TENEX. The way this works is that the user, after typing the beginning of a word, hits the "recognize" key. If there is a unique command that the system knows about which starts with what has been typed, the system finishes typing that word for the user. Otherwise, it can simply refuse by typing back a bell (as is done in TENEX), or it can guess. In a system for children, the computer would probably keep track of all the commands a specific child already has defined or has used successfully, as opposed to recognizing all commands that the system knows about, to minimize the number of

commands in the list and therefore minimize the numbers of characters the child has to type.

If there is a pointing device available, such as a mouse or a light pen, the system can display the possible choices for commands, and the user can point to the correct one. If the child cannot read, the commands can be displayed as pictures. If there are too many legal commands to fit on the screen, they can be grouped under headings such as "music commands" or "turtle commands", and when the user points to a group, then the list of commands within that group gets displayed. If the system keeps track of what the child has typed, it can guess at a subset of commands that the child will want to use next. For instance, if the child has just typed a command that needs an argument, the system knows the next thing to be typed must be either a number, a variable, or a function. If the system only displays commands that would be legal in that context, the user will never have syntactic errors.

In a computer language such as Lisp, where the user is forced to put parentheses everywhere, there is no ambiguity in what is typed. But Lisp code, although easy for the computer to read, is unreadable by humans, since it is hard to see how the parentheses match up. In a language like Smalltalk, although the code is not littered with parentheses, it is still hard for humans to read the code, sometimes because an expression can be parsed in two different ways (if the user is not familiar with the parsing algorithm), sometimes because it is not obvious what words are variables and what are commands, and sometimes for other reasons. If the system parses the expression as the user types it in, it can group arguments with the calling function, clearly separate commands, and display the program in a readable form. If there are ambiguities the user will see how the system intends to group things, and the user can correct them if he likes.

Most standard editors are hard to use, since the user has to move around a "cursor" with magic incantations. With a display and a pointing device, the user can point to the location he wants to change, and the change will occur as he watches, so he will know exactly what he has done.

For young children especially it is important to display the execution of a program, so they can see just what the different commands do. This would serve as an effective debugging aid, but it is important not to display so much information that the user gets swamped, or so little as to miss exhibiting the information the user needs. There is still a lot of work to be done on debugging aids for higher level languages.

Because a system designed for young children will be one that works on minimizing the amount of work the user must do to get an effect, many of the ideas for this system can be used for physically handicapped people. With some work on input devices for the physically handicapped, a very usable system could be implemented.

In addition to work on expanding and improving the environment, there is a whole area of research involving working with children with the system, finding difficulties they encounter, and either inventing new ways to present the concepts they find difficult or deciding that there is some maturing the child needs in order to gain certain concepts.

Bibliography

Bamberger, Jeanne, The Luxury of Necessity, Logo Memo 12, MIT, Cambridge, Mass., 1974.

Bamberger, Jeanne, What's in a Tune, Logo Memo 13, MIT, Cambridge, Mass., 1974.

Goldenberg, Paul, A Glossary of PDP11 Logo Primitives, Logo Memo 16, MIT, Cambridge, Mass., 1975.

Kay, Alan, The Reactive Engine, Ph.D. thesis, University of Utah, Salt Lake City, 1969.

Kay, Alan, "A Personal Computer for Children of All Ages", Proc. ACM National Conference, Boston, 1972.

Papert, Seymour, A Computer Laboratory for Elementary Schools, AI Memo 246, MIT, Cambridge, Mass., 1971.

Papert, Seymour, Teaching Children Thinking, AI Memo 247, MIT, Cambridge, Mass, 1971.

Papert, Seymour, Uses of Technology to Enhance Education, AI Memo 298, MIT, Cambridge, Mass, 1973.

Perlman, Radia, TORTIS--Toddler's Own Recursive Turtle Interpreter System, Logo Memo 9, MIT, Cambridge, Mass, 1974.

Smith, David, PYGMALION: A Creative Programming Environment, AI Memo 260, Stanford Artificial Intelligence Laboratory, Stanford, Calif., 1975.

Swinehart, Daniel, COPILOT: A Multiple Process Approach to Interactive Programming Systems, AI Memo 230, Stanford Artificial Intelligence Laboratory, Stanford, Calif., 1974.

Weir, Sylvia and Emanuel, Ricky, Using Logo to Catalyse Communication in an Autistic Child, DAI Research Report 15, Department of Artificial Intelligence, University of Edinburgh, 1976.